US 20120144195A1

(19) **United States**

(12) **Patent Application Publication**

Nair et al.

(10) **Pub. No.: US 2012/0144195 A1**

(43) **Pub. Date: Jun. 7, 2012**

(54) **METHOD AND SYSTEM FOR UNIFIED MOBILE CONTENT PROTECTION**

(75) Inventors: **Raj Nair**, Lexington, MA (US); **Mikhail Mikhailov**, Newton, MA (US)

(73) Assignee: **AZUKI SYSTEMS, INC.**, Acton, MA (US)

(21) Appl. No.: **13/370,537**

(22) Filed: **Feb. 10, 2012**

Related U.S. Application Data

(63) Continuation of application No. PCT/US2010/045596, filed on Aug. 16, 2010.

(60) Provisional application No. 61/234,092, filed on Aug. 14, 2009.

Publication Classification

(51) **Int. Cl.**

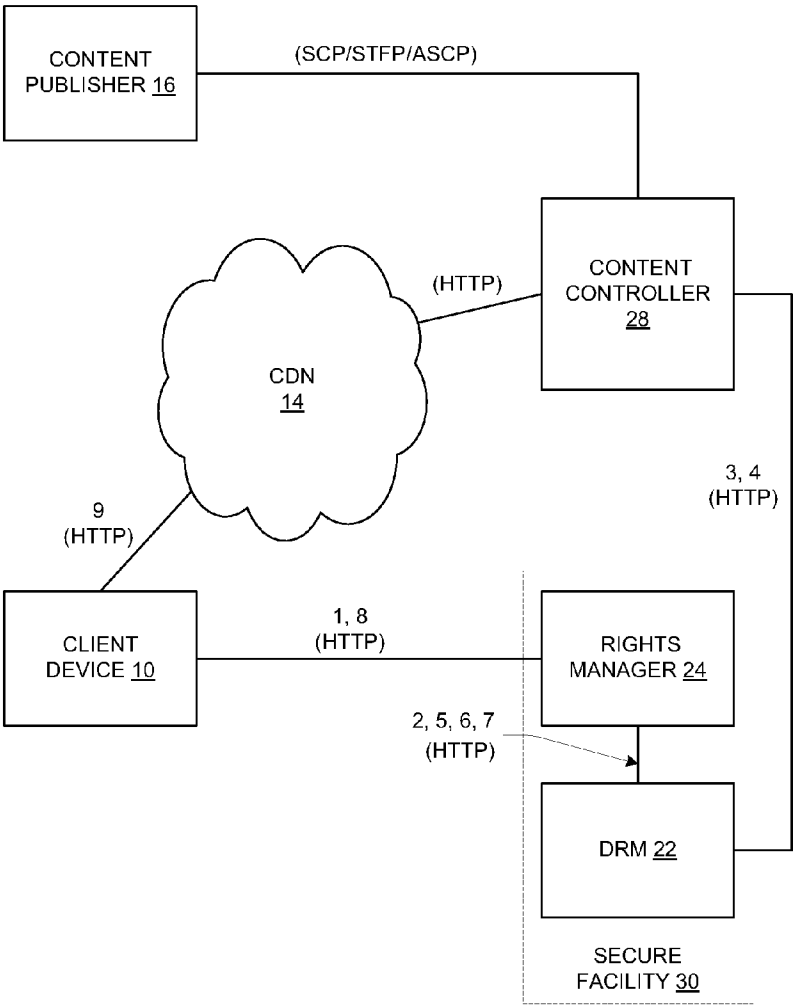
G06F 21/24 (2006.01)

H04L 9/32 (2006.01)

(52) **U.S. Cl.** **713/168**; 726/29

(57) **ABSTRACT**

Media content is delivered to a variety of mobile devices in a protected manner based on client-server architecture with a symmetric (private-key) encryption scheme. A media preparation server (MPS) encrypts media content and publishes and stores it on a content delivery server (CDS), such as a server in a content distribution network (CDN). Client devices can freely obtain the media content from the CDS and can also freely distribute the media content further. They cannot, however, play the content without first obtaining a decryption key and license. Access to decryption keys is via a centralized rights manager, providing a desired level of DRM control.



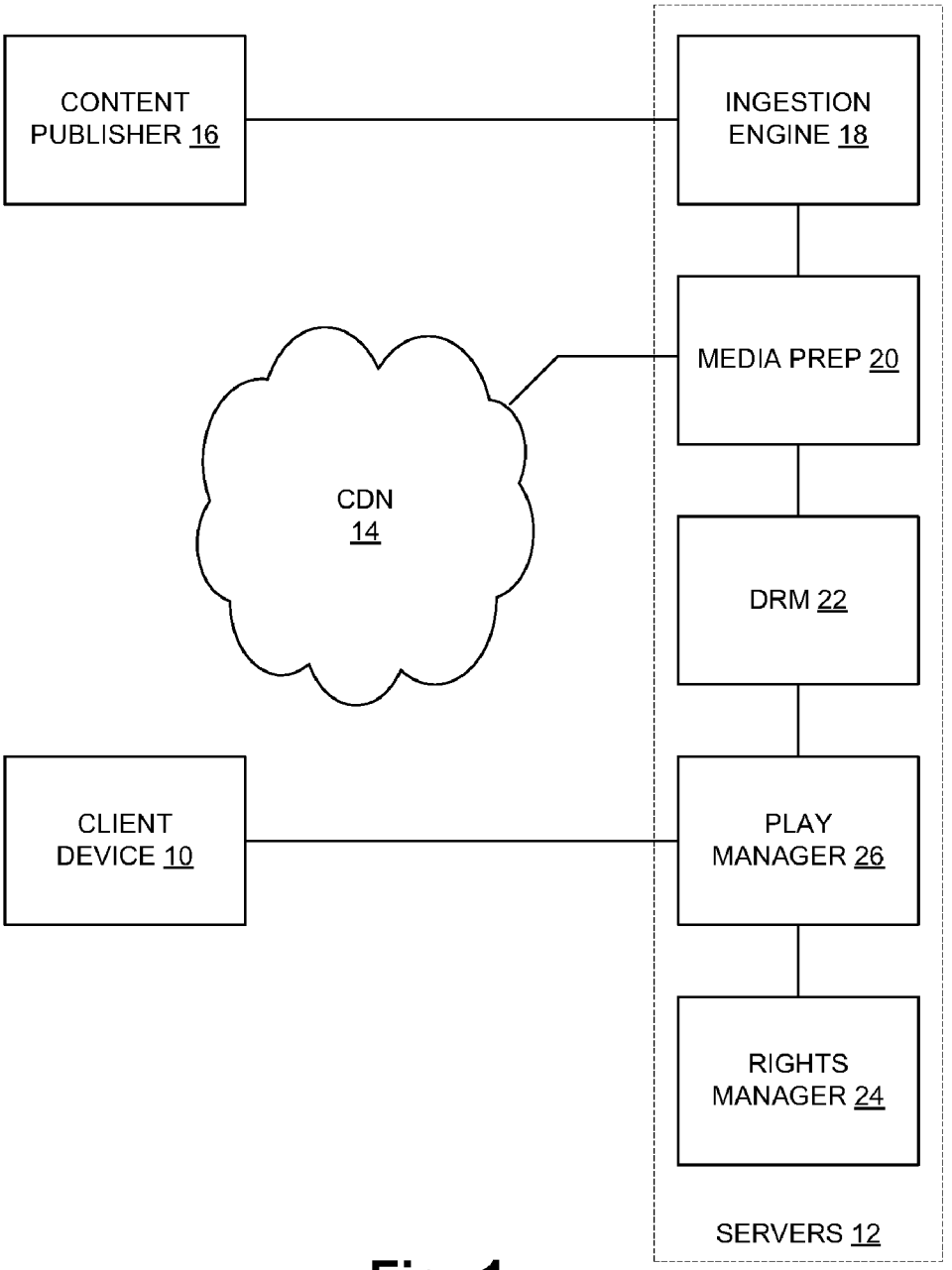

CONTENT PUBLISHER 16
INGESTION ENGINE 18
MEDIA PREP 20
CDN 14
DRM 22
CLIENT DEVICE 10
PLAY MANAGER 26
RIGHTS MANAGER 24
SERVERS 12

Fig. 1

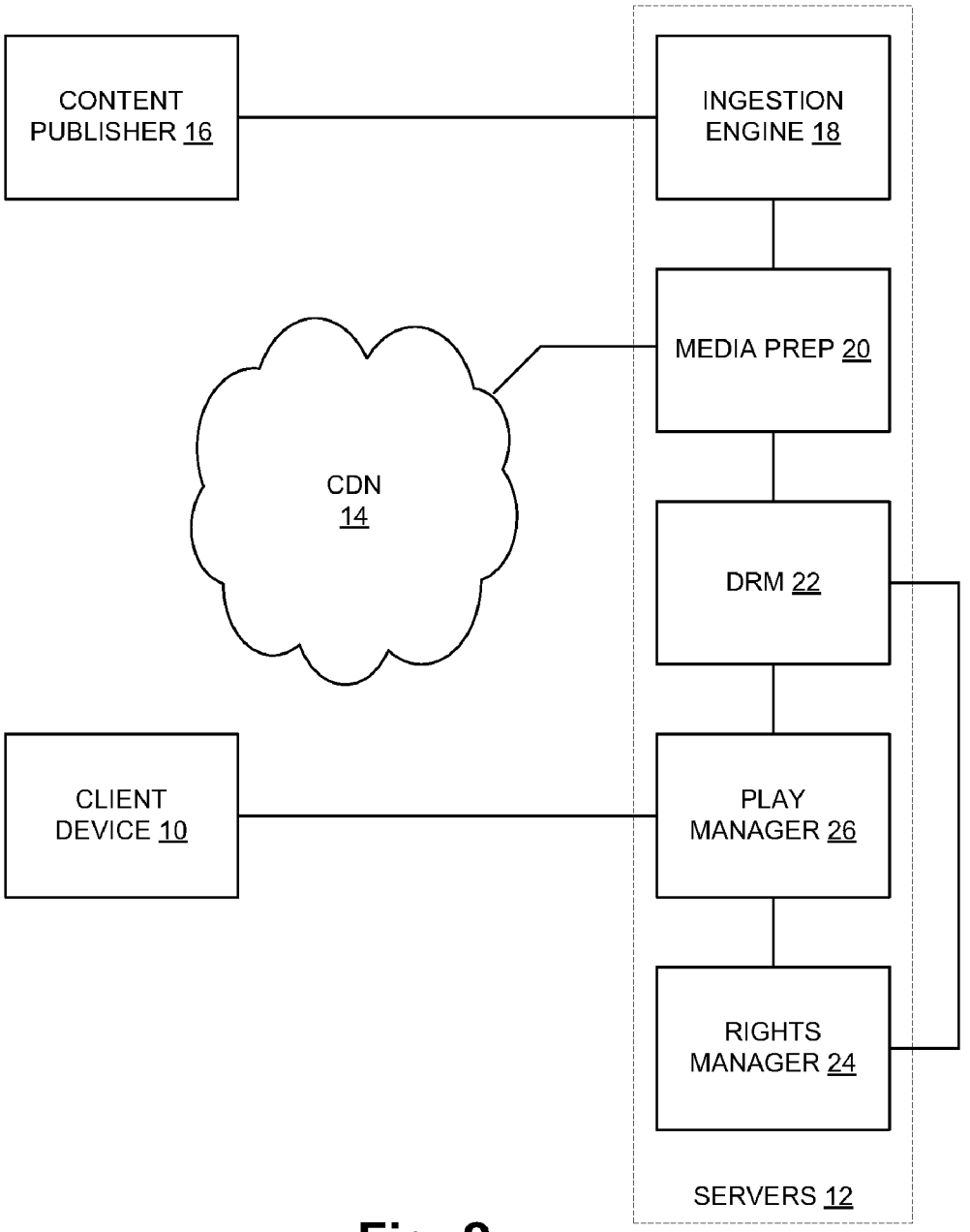

CONTENT PUBLISHER 16
INGESTION ENGINE 18
MEDIA PREP 20
CDN 14
DRM 22
CLIENT DEVICE 10
PLAY MANAGER 26
RIGHTS MANAGER 24
SERVERS 12

Fig. 2

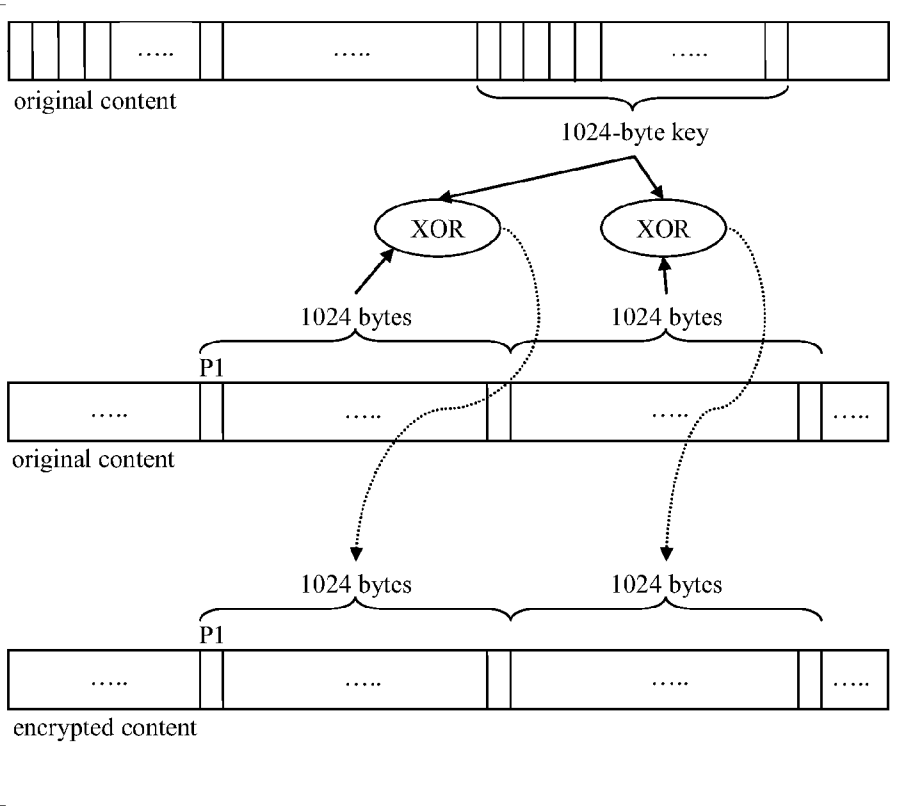

original content
1024-byte key
XOR
XOR
1024 bytes
1024 bytes
P1
original content
1024 bytes
1024 bytes
P1
encrypted content

Fig. 3

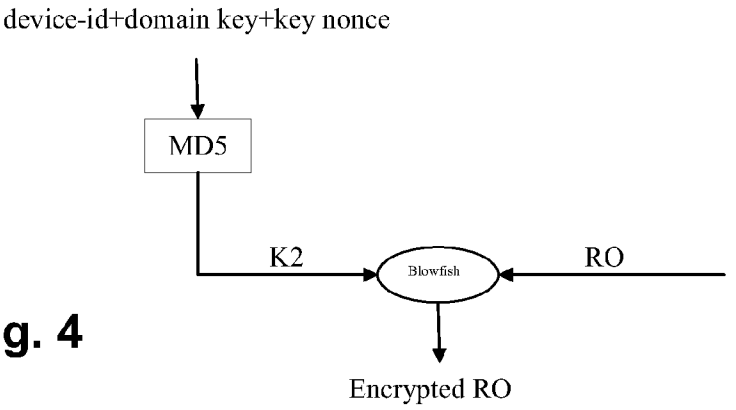

device-id+domain key+key nonce
MD5
K2
Blowfish
RO
Encrypted RO

Fig. 4

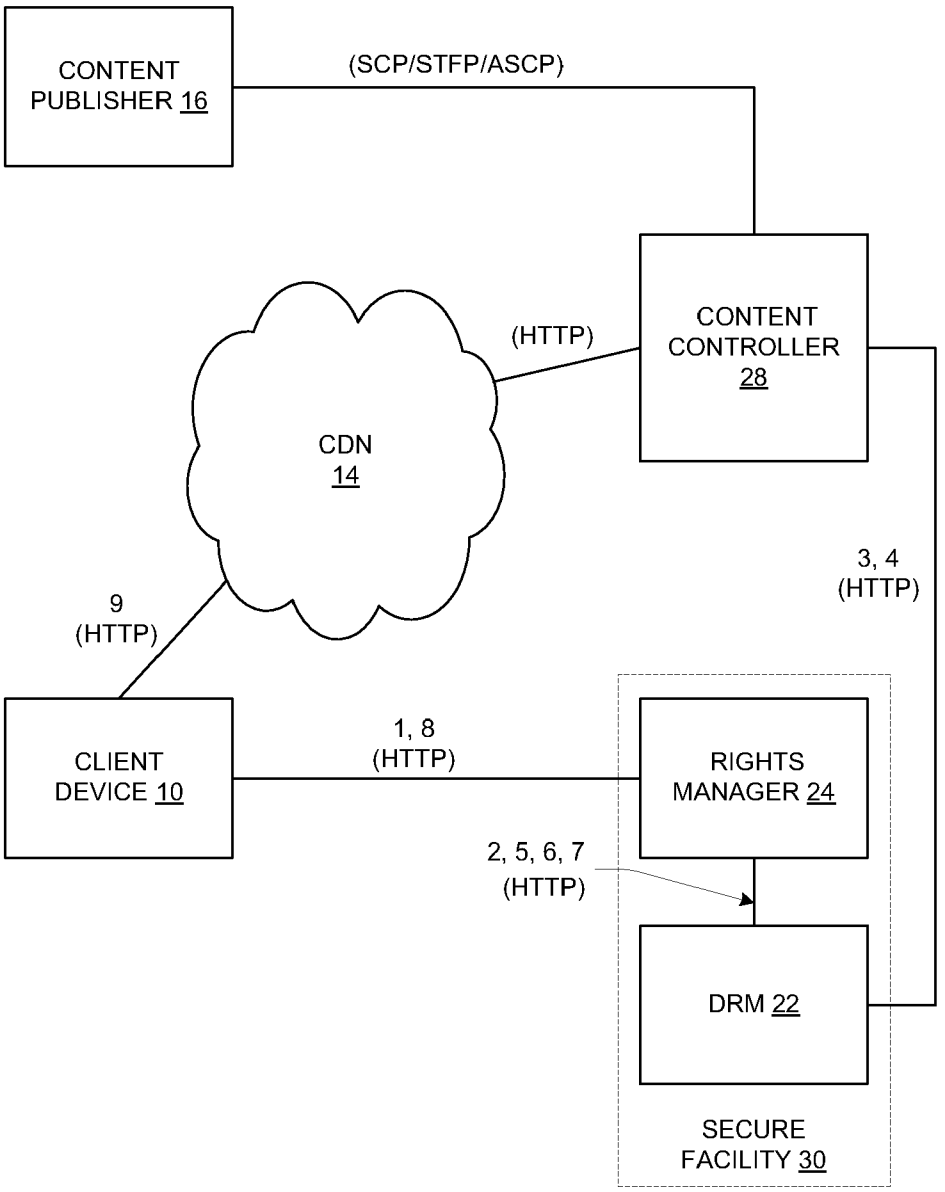

CONTENT PUBLISHER 16
(SCP/STFP/ASCP)
CONTENT CONTROLLER 28
(HTTP)
CDN 14
3, 4 (HTTP)
9 (HTTP)
CLIENT DEVICE 10
1, 8 (HTTP)
RIGHTS MANAGER 24
2, 5, 6, 7 (HTTP)
DRM 22
SECURE FACILITY 30

Fig. 5

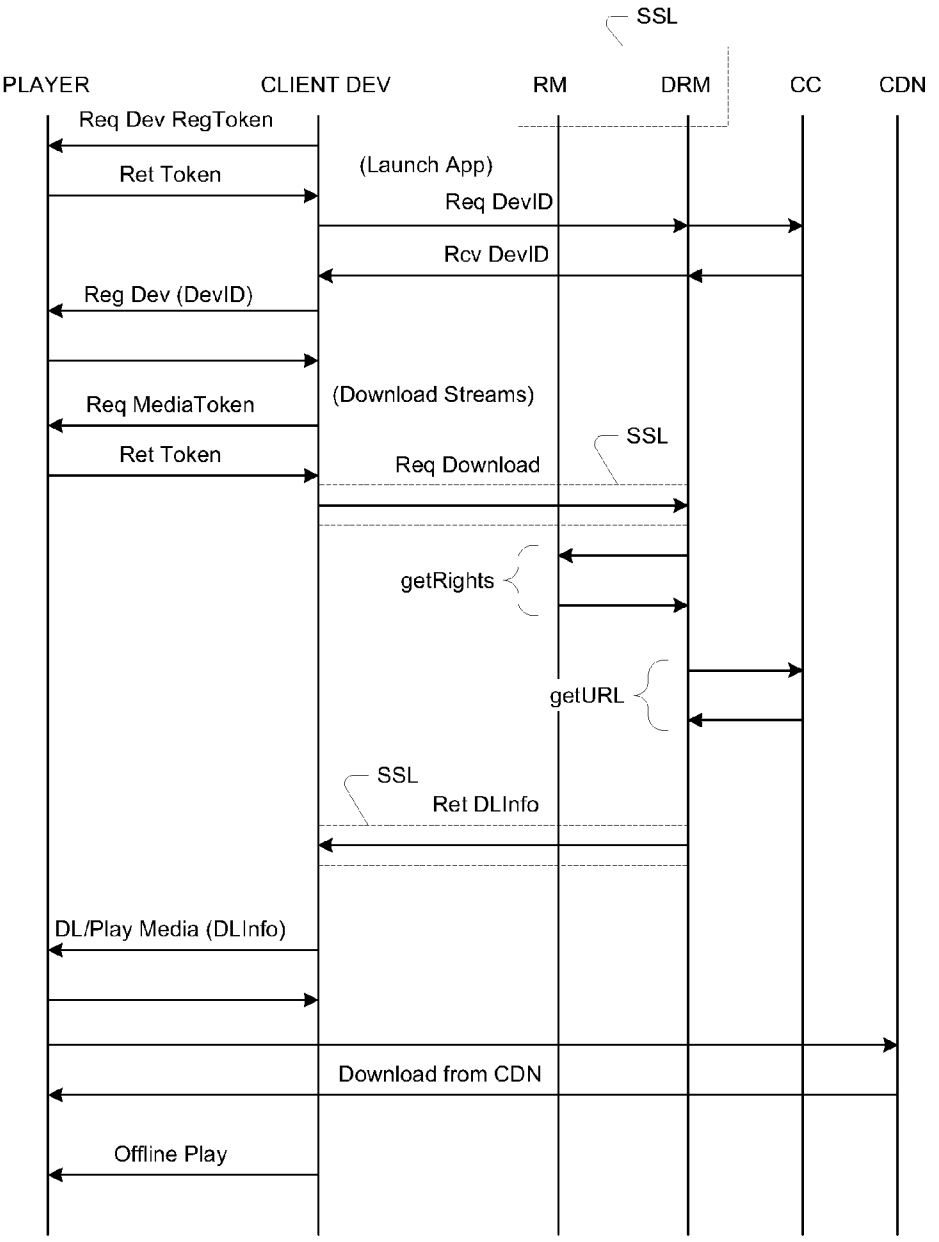

SSL
PLAYER
CLIENT DEV
RM
DRM
CC
CDN
Req Dev RegToken
Ret Token
(Launch App)
Req DevID
Rcv DevID
Reg Dev (DevID)
(Download Streams)
Req MediaToken
Ret Token
SSL
Req Download
getRights
getURL
SSL
Ret DLInfo
DL/Play Media (DLInfo)
Download from CDN
Offline Play

Fig. 6

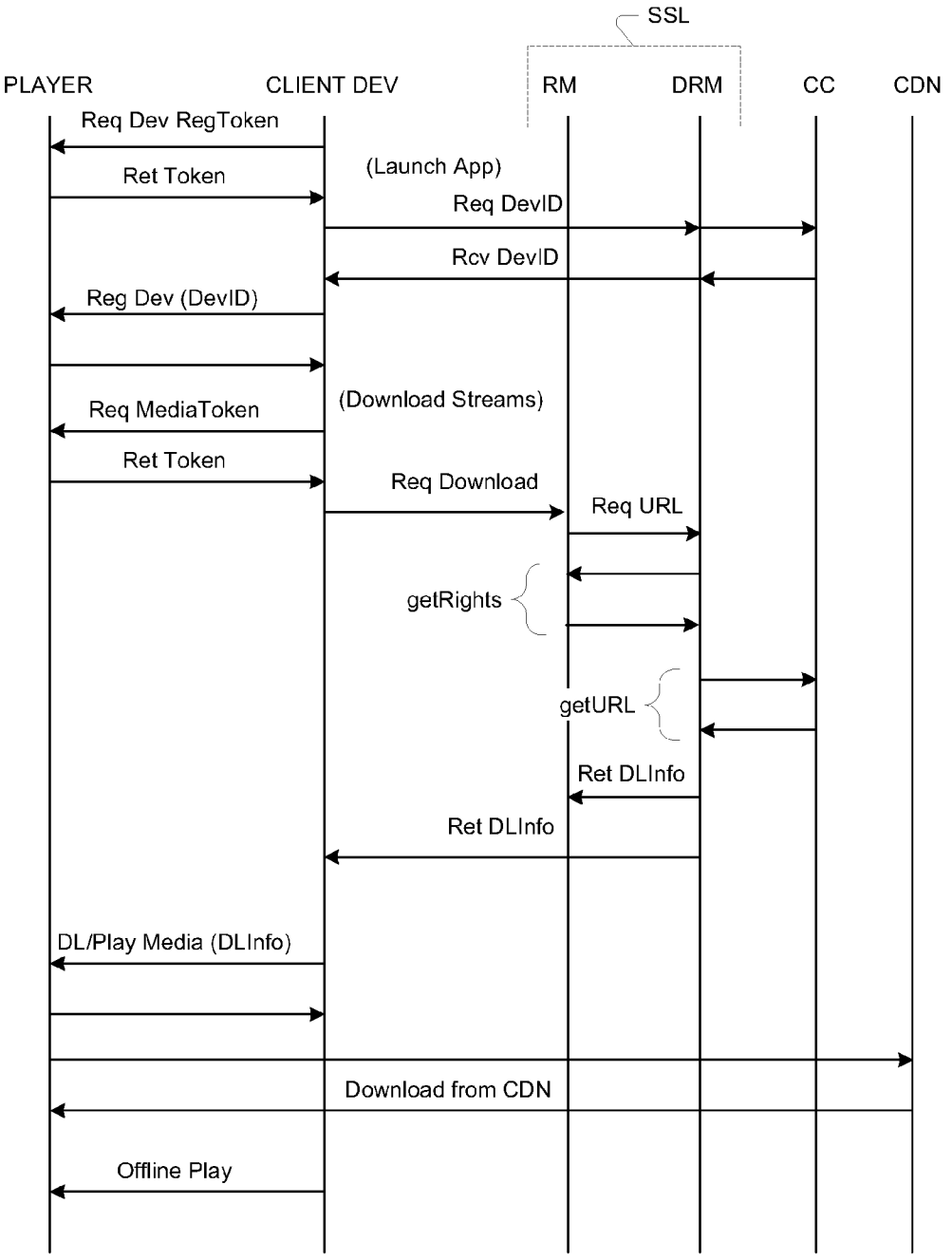

SSL
PLAYER
CLIENT DEV
RM
DRM
CC
CDN
Req Dev RegToken
Ret Token
(Launch App)
Req DevID
Rcv DevID
Reg Dev (DevID)
(Download Streams)
Req MediaToken
Ret Token
Req Download
Req URL
getRights
getURL
Ret DLInfo
Ret DLInfo
DL/Play Media (DLInfo)
Download from CDN
Offline Play

Fig. 7

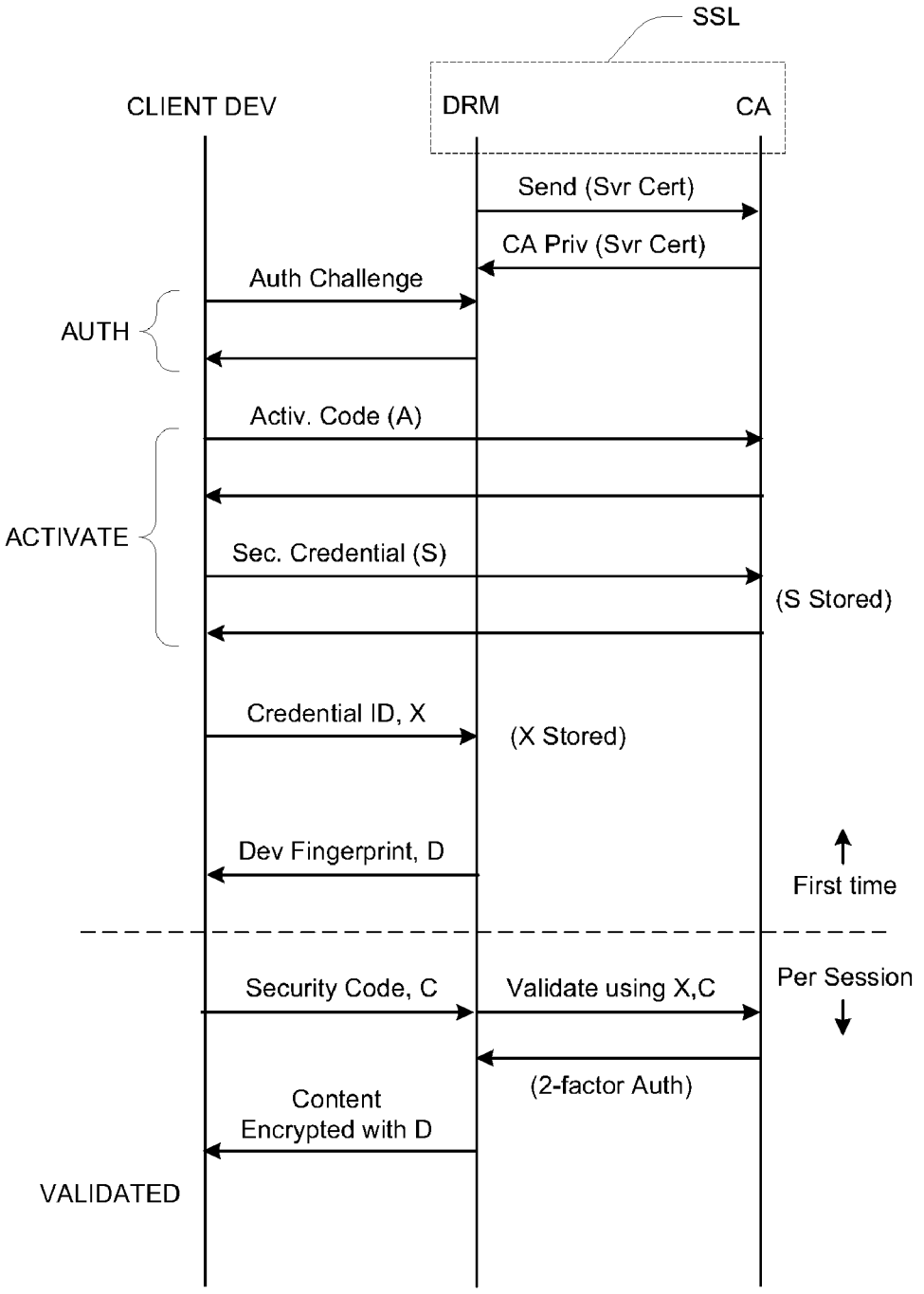

SSL
CLIENT DEV
DRM
CA
Send (Svr Cert)
CA Priv (Svr Cert)
Auth Challenge
AUTH
Activ. Code (A)
ACTIVATE
Sec. Credential (S)
(S Stored)
Credential ID, X
(X Stored)
Dev Fingerprint, D
First time
Security Code, C
Validate using X,C
Per Session
(2-factor Auth)
Content
Encrypted with D
VALIDATED

Fig. 8

METHOD AND SYSTEM FOR UNIFIED MOBILE CONTENT PROTECTION

BACKGROUND

[0001] The growing number of large form factor mobile devices such as the iPad has revolutionized mobile media consumption leading to revolutionary initiatives such as "TV Everywhere™" (TVE) with a mandate to make premium content available on a wide range of devices with great diversity in capabilities. This type of distribution, sometimes known as "Over-The-Top" (OTT) distribution, has underscored the need for a new and more robust trust model that builds on a 2-part trust model of user authentication and device identification and can offer the same level of content protection that content owners have had in the closed Consumer Electronics ecosystems of the past. The added level of protection can enable publishers to fully realize the potential for content distribution through this new open ecosystem of devices.

[0002] Content protection is challenging in mobile devices for a number of reasons. Mobile devices do not uniformly support Digital Rights Management (DRM) standards. In particular, most mobile devices do not currently support the most comprehensive form of content protection, the Open Mobile Alliance (OMA) V2.0 DRM standard. Mobile devices also vary in their CPU performance and memory capacity. An additional complication is the need to support multiple modes of delivery required in the mobile environment, such as live streaming, watching short video segments, rentals, or media download for watching later.

[0003] Current media protection schemes depend on sending the license information in-band with the media or using a pre-distributed license key in the media viewing device. Examples are Playready, WMDRM, Widevine, and Flash Access. However, TVE requires that the rights are transferable across devices in a seamless manner.

SUMMARY

[0004] The present invention relates in general to protecting media on mobile devices and more specifically to implementing a content protection system for media that may be streamed or watched offline on mobile devices. This system is particularly useful in the deployment of TVE services for protecting media on any Internet-connected device in an "Over-The-Top" (OTT) manner where the digital rights to the media are delivered to the device over the network and made specific to the device and user.

[0005] Methods and apparatus are disclosed for protecting content delivered to a variety of mobile devices based on client-server architecture with a symmetric (private-key) encryption scheme. In one embodiment, a media preparation server (MPS) encrypts all media content and publishes and stores it on a content delivery server (CDS), such as a server in a content distribution network (CDN). Clients can freely obtain media content from the CDS and can also freely distribute it further. They cannot, however, play the content without first obtaining a decryption key. Access to decryption keys is via a centralized rights manager, providing a desired level of DRM control.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The foregoing and other objects, features and advantages will be apparent from the following description of particular embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of various embodiments of the invention.

[0007] FIGS. **1** and **2** are block diagrams of systems capable of conducting procedures, in accordance with various embodiments of the invention;

[0008] FIG. **3** is a diagram of the content encryption, in accordance with an embodiment of the present invention;

[0009] FIG. **4** is a diagram of the key wrapping, in accordance with an embodiment of the present invention; and

[0010] FIG. **5** is a block diagram of a system showing interfaces and message formats between system components;

[0011] FIGS. **6-8** are diagrams of message flows during system operation in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0012] FIG. **1** is a block diagram for one embodiment of the present invention. It shows a client device **10** and a plurality of servers **12** that are connected together in a secure network and together form an instance of what is referred to herein as a "wireless platform" (WP). The client device **10** and WP servers **12** are typically computerized devices which include one or more processors, memory, storage (e.g., magnetic or flash memory storage), and input/output circuitry all coupled together by one or more data buses, along with program instructions which are executed by the processor out of the memory to perform certain functions which are described herein. Part or all of the functions may be depicted by corresponding blocks in the drawings, and these should be understood to cover a computerized device programmed to perform the identified function.

[0013] In one embodiment, the servers **12** (referred to as servers herein) may be collocated in a single data center. In another embodiment, the servers **12** may be geographically distributed in multiple data centers. In another embodiment, the servers **12** may be physically in the same region, but connected to the client **10** through separate network paths (e.g. through different network service providers). In one embodiment, the servers **12** are situated as part of a content delivery network (CDN) **14**. In one embodiment, the content from a content publisher **16** is ingested via an ingestion engine **18**, and the ingested content is then segmented by a media preparation engine (MEDIA PREP) **20**. The media preparation engine **20** obtains a content encryption/decryption key from a digital rights management (DRM) server **22** and uses it to encrypt the content for storage and later delivery in encrypted form. An example of streaming of content is shown in published PCT application WO 2010/045109.

[0014] In part of the description below, the combination of the ingestion engine **18**, media prep **20** and a play manager **26** are referred to as a "content controller". Thus in one embodiment the system is constituted by a content controller along with a DRM server **22** and a rights manager **24**.

[0015] A media preparation profile in the media preparation server **20** specifies an encryption type on a per-media-item basis. Candidate ciphers may include XOR, RC4, HC-128, AES, and along with the specification of encryption type is stored a corresponding key and a key length. Each media item has its own randomly generated key value. In the case of AES and XOR encryption, this randomly generated

key value is used as the actual key for encryption/decryption, whereas for RC4 and HC-128 it is the seed key to initialize a stream cipher. AES key length is typically 128 bits. XOR key length may be 1024 bytes, and may be configurable. RC4 and HC-128 use 128 bit seed keys. The media preparation profile also specifies, on a per-media basis, the length of the byte stream which should be generated (this is the actual key used for media encryption, and the length is the same as the block length described elsewhere herein). Each media item is transcoded in multiple formats for different target platforms, and each of the resulting transcoded media files is immediately encrypted with the chosen cipher and key and the encrypted files are then pushed to the CDN **14**. Additional details regarding encryption are provided below.

[0016] In order to use the system for downloading content, the client device **10** first authenticates with the rights manager **24** and registers its device with the DRM server **20**. During this process the client **10** obtains a logical device id from the rights manager **24** that is a token to represent a user of the client device **10**, and associates this token with the specific client device **10** via a device "fingerprint" which is a unique identifier for the client device **10**. The unique identification may be based on certain physical properties that may include an international mobile equipment identifier (IMEI) number, media access control (MAC) address, or certain file system properties. Each of the supported client devices **10** provides an Application Programming Interface (API) via which the unique identifier of that device can be obtained. Some devices have an IMEI number, some a mobile equipment identifier (MEID) number, some an electronic serial number (ESN). The iPhone has a unique device identifier (UDID).

[0017] The client device **10** has a built-in domain key that is used to encrypt the exchange of the logical device ID with the rights manager **24**. For enhanced security, the domain key is divided into a number of separate components which are stored so as to be difficult to locate. For example, each may be stored as an array, with elements of the array represented as strings containing an integer and some special characters which are commonly found in binary files. When the arrays are examined, it is difficult to detect where the components are located. At run-time, all arrays are processed, special characters are discarded, and elements of these arrays are converted into characters and concatenated together to produce the actual domain key.

[0018] Device registration is carried out as follows. A DRM Agent running on the client **10** generates an encrypted token containing a device id and a randomly generated long nonce. That information (device id and the random long nonce) is encrypted with the domain key and is sent to the DRM server **22**. The DRM server **22** decrypts this registration message using the same domain key, and stores an association between this user, device id, and key nonce in a database. The DRM server **22** generates a response containing a unique logical id assigned to this user. This response is encrypted with a session key constructed from domain key, device id, and the random key nonce provided by the DRM Agent, and the encrypted response is sent to the client **10**. Details regarding the construction of the session key are provided below.

[0019] Once the client **10** receives the response, it decrypts the response and stores registration information into an encrypted rights file on the client device **10**. The rights file is encrypted with the key constructed from the combination of the domain key and the device id.

[0020] The session key may be constructed as follows:

[0021] A shared or domain key is combined with the device id and the randomly generated key nonce: shared_key+device_id+key_nonce. The resulting string is fairly long, so a hash or checksum is computed on it. In one embodiment, a hex representation of the hash, which may be 32 bytes long, is chosen to be the key. In different embodiments, the raw hash output (which may be a 128-bit integer) may be used. In one embodiment a Message Digest **5** (MD**5**) hash may be used. Other embodiments might use a 64-bit RACE Integrity Primitives Evaluation Message Digest (RIPEMD) hash function instead of MD**5**.

[0022] In other embodiments, it is possible to include other individualization parameters into these keys. Thus, the client/server session key, as well as the key used to encrypt the rights file on the device, could be enhanced further by adding unique user information (user token) and/or application information, such as the application name or identifier. That way, keys will be application-specific and user-specific as well as device-specific.

[0023] FIG. **2** shows a slightly different embodiment of the system, in which there is a connection directly between the rights manager **24** and the DRM server **22** to enable the DRM server **22** to directly consult with the rights manager **24** as may be required.

[0024] As previously mentioned, any of various content encryption schemes may be employed. The following presents several specific examples along with corresponding details regarding how encryption/decryption is carried out. In some embodiments, the encryption can be applied to portions of the file such as key frames for video in order to reduce processing load.

XOR

[0025] In one embodiment the following simple and fast symmetric (private key) encryption scheme is used. Operation is illustrated in FIG. **3**. The media preparation server **20** performs an exclusive-OR operation (XOR) between the contents of the media file and a secret (private) key K**1** (not shown). In one embodiment the key K**1** is 1024 bytes long. The XOR operation starts at a random position P**1** within the media file and continues until the end of the file. The random position P**1** is preferably chosen to be close to the beginning of the file, e.g. within the first 10% of the file. P**1** can also be a predetermined fixed position, for example the very beginning of the file (location 0).

[0026] The key K**1** may be chosen in a variety of ways. For example, it may be generated randomly. Alternatively, it may be generated by first choosing another random position P**2** (not shown) within the same file, and selecting 1024 bytes from the media file starting at position P**2**. If there are not 1024 bytes remaining between P2 and the end of the file, then 1024 bytes are selected starting at position P**2**−1024+1. As noted, the key length may be other than 1024 bytes, and may be configurable.

[0027] The media preparation server **20** stores P**1** and P**2** in a database for each media file. In addition, the media preparation server **20** associates an expiration time with the encryption keys, stores the expiration time in the database, and re-encrypts content with new keys upon key expiration.

RC4-Drop(n)

[0028] RC4-drop(n) is a stream-cipher algorithm generally known to those skilled in the art. It includes the dropping of

the first 3072 bytes from each generated keystream. Also, RC4 does not have a formal notion of an initialization vector (IV). Instead, a checksum is computed on a concatenated key and an arbitrarily chosen initialization value, and the checksum is used as the key.

[0029] In one embodiment of stream cipher encoding, the entire media file is divided into smaller blocks of a selected block size. With a stream cipher, one can generate an infinitely-long stream of bytes. Theoretically, if a content item (e.g., movie) were to be played only from start to finish, without rewinding or fast-forwarding (i.e. without scrubbing), a stream cipher could be used on the streaming media without specialization. However, since the user may scrub during playback, decryption requires a modification to the stream cipher. The media is divided into fixed-size blocks and a new stream of key bytes is generated for each block by using the same seed key and a different IV. The IV in this case can be just the sequential block number, starting from 0. In one embodiment the blocks can have length 32 k, but the block length can be different in other embodiments and may be configurable.

HC-128

[0030] HC-128 is another well-known stream cipher whose block size can be adapted as described above. Also, in addition to block size, both RC4 and HC-128 can take into account a segment number for live streaming and for video on demand (VOD). The entire long-form content is represented as many segments, and each segment is then divided into multiple blocks from the encryption/decryption point of view.

AES

[0031] The same approach to block sizing may be taken for AES unless of course in some embodiments the decryption is done in hardware. It may be desirable to use the same form of AES encryption supported by iPhone® and iPad®, which is AES bit with Cipher-Block-Chaining (CBC mode). Each segment is encrypted individually, and the same key is used across all segments, but each segment has its own initialization vector which is the sequence number of the segment.

[0032] It is briefly described how a user obtains a rights object (RO) to use in downloading and streaming, as well as playing, content. The user registers with a content provider using, in one embodiment, OpenID technology and obtains a user token which uniquely identifies that user. Before the user can play a given media content file, the user must obtain the decryption key. A DRM agent running on the client device **10** contacts the rights manager **24** and provides three items: <device-id, media-id, user-token>, where device-id is a unique identifier specific to that particular mobile device, media-id is a unique identifier specific to the particular media content the user wants to play, and user-token is the unique user identifier. Device id could be the unique address of the mobile device, or it may be one of the types of device identifiers discussed above.

[0033] The rights manager **24** receives the request for the RO from the client **10**, containing <device-id, media-id, user-token>. The rights manager **24** validates the user-token using OpenID technology and also validates that media-id is correct and has not expired. It then generates the requested RO, which contains a key value **K1** for media content decryption, a remaining play count for that media, and a media license expiration time. Even though communications between the client **10** and rights manager **24** is carried over a secure connection (SSL), the rights manager **24** may optionally encrypt the RO so that encrypted RO can be safely stored on the client device **10**.

[0034] The encryption of the RO is illustrated in FIG. **4**. To encrypt the RO, the rights manager **24** uses the following symmetric encryption scheme. The RO is encrypted with 64-bit Blowfish key constructed from the checksum (domain key+device id+key nonce). To compute **K2**, the rights manager **24** applies an MD**5** checksum function to the device-id.

Message Flow

[0035] FIG. **5** contains a block diagram showing interfaces (numbered reference points) between system components. The following is a description of messaging flows (including message formats) among the components.

[0036] The ingestion flow consists of secure transfer of content from the content publisher **16** via a secure transfer method such as scp, sftp, or ascp (Aspera) to the content controller back-end server **28**, which in turn transcodes and encrypts the content using the chosen content cipher (e.g., AES or HC-128) and publishes it into the CDN **14**.

[0037] The interfaces and associated protocols are described next for each of the numbered reference points in FIG. **5**.

[0038] 1. Over this HTTP interface, the client **10** performs a one-time device registration with the rights manager **24** (and DRM Server **22**) passing the device-id, key nonce, and message nonce encrypted with the Blowfish algorithm using the domain key that is stored in an obfuscated manner in the application binary as described above. The registration information is passed through to the DRM Server **22** via the interface **2** described below. Depending on specific deployment requirements, the client may **10** may alternatively go to the DRM server **22** first, and the DRM server **22** then communicates with the rights manager **24**.

[0039] Also, on the same interface, every time the client **10** needs to play a media, it sends media rights requests to the rights manager **24** also encrypted via Blowfish with a device-specific key. The media rights request contains device id, media id, logical id (a unique abstract user identifier) provided by the DRM server **24** when the device was registered, message nonce, and the current play count.

[0040] 2. This HTTP interface is used as a pass-through interface, where the rights manager **24** relays requests (device registration and media location and rights requests) received from the client **10** and destined to the DRM server **22**. These messages are encrypted as noted in #1. The rights manager **24** maintains user information which the DRM server **22** does not have access to, and the rights manager **24** maps individual users to logical ids maintained by the DRM server **22**. The rights manager **24** appends the logical id, uniquely identifying the current user, to all requests being forwarded to the DRM server **22**. The only exception is the initial device registration because it does not have a logical id for that user at that point. The logical ids need not be encrypted when these servers are in a secure facility **30** with restricted access as shown. In environments where these servers need to be remote, a secure connection would be needed between them. The secure connection may take the form of a virtual private network (VPN) or a Secure Sockets Layer (SSL) connection.

[0041] 3. This HTTP interface is used by the DRM server **22** to request media information from the back-end content controller **28**. This interface is used to obtain information

needed to play a media item. The request by itself does not have any commercial value and is therefore not encrypted nor sent over a secure channel.

[0042] 4. This HTTP interface carries the response of the content controller **26** to the DRM server request described under item #3 above. The response is an XML document, containing media URL pointing to an encrypted media file located in the CDN **14** and an encrypted message which contains information about the cipher and the key used to encrypt this media. The message is encrypted with the Blowfish algorithm and the domain key.

[0043] 5. Via this HTTP interface the DRM server **22** asks the rights manager **24** for media rights for the current user. The request contains logical id, media id, and the play count reported by the client **10**. This interface is used when the client **10** is requesting media rights as described in #1. The information need not be encrypted when the DRM server **22** and rights manager **24** are in a secure facility **30**. Alternatively, a secure connection may be employed.

[0044] 6. This HTTP interface carries the response of rights manager **24** to the DRM server request described under item #5 above. The response is an XML document containing rights information for the requested media and the current user. This interface is used only when the client **10** is requesting media rights. The response need not be encrypted when the DRM server **22** and rights manager **24** are in a secure facility **30**. Alternatively, a secure connection may be employed.

[0045] 7. This HTTP interface sends the response of the DRM server **22** to the rights manager **24**. Two types of responses are sent over this interface: the device registration response and the media location and rights in response to requests described under item #2 above.

[0046] The device registration response is an XML document that contains an encrypted message (containing the logical id) destined for the client **10**, and also the logical id and total device count for the current user in the clear. The rights manager **24** uses the device count to check against the total count of authorized devices for the user. It removes the logical id and device count from the response, before forwarding it to the client **10** on interface **8**. The client completes the registration on its end when it can receive the encrypted message and successfully decrypt and verify the nonce and checksum in the message.

[0047] The media rights and location response is an XML document that contains the media URL pointing to an encrypted media file located in a CDN **14**, and an encrypted message (destined for the client) which contains information about the cipher and the key needed to decrypt this media and media rights information for the current user. This response is forwarded to the client **10**.

[0048] In both types of responses, the message is encrypted with a key produced from the domain key, device id, and the key nonce.

[0049] 8. This is a pass-through interface where the rights manager **24** simply forwards the responses it received from the DRM server **22** to the client **10**, in response to the client's requests described under item #1 above. The contents of these responses are described fully in #7.

[0050] 9. This is the interface by which the content is delivered to the client **10** from the CDN **14**.

[0051] FIG. **6** is a message flow ladder-diagram for one embodiment of the present invention. It describes the message flow for device registration and obtaining the rights object containing the content key for playing the content. FIG. **7** contains another message flow ladder-diagram for an alternate embodiment where the client **10** is in direct communication with the intervening rights manager **24**, which in turn communicates with DRM server **22**.

[0052] FIG. **8** illustrates a work flow for integrating functionality of a certificate authority (CA) into the content protection scheme. The work flow consists of the following steps:

[0053] 1. Have a server certificate signed by the CA

[0054] 2. Distribute the application to devices via application stores

[0055] 3. Initially, a client **10** authenticates with a server via SSL via the following:

- **[0056]** A. Authentication could be passed through to a customer authentication server **12**
- **[0057]** B. The customer authentication server **12** receives an activation code from the CA and passes it to the client **10**
- **[0058]** C. The client **10** uses the activation code (using tools of a software development kit (SDK) of the CA) to obtain an encrypted security credential from the CA, wherein the security credential=(the shared secret, a credential ID, and a creation time)
- **[0059]** D. The client **10** sends the credential ID to the server **12** which is linked to an authentication record
- **[0060]** E. The client **10** registers by sending the device fingerprint to the server **12** and gets a device-specific key

[0061] 4. For a session, a client **10** is validated as follows:

- **[0062]** A. The client uses the CA SDK to dynamically generate a security code from the security credential and sends it to the server **12** via SSL
- **[0063]** B. The server **12** contacts the CA to validate the client **10** using the stored credential ID together with the security code
- **[0064]** C. The server **12** returns the content key encrypted with the device-specific key
- **[0065]** D. In offline mode, the DRM agent of the client **10** will offer protection with an offline timeout that forces contact with server **12** (which includes protection against clock tampering as described below)

Anti-Clock Rollback Protection

[0066] Clock rollback is a technique employed to illegally extend time-based licenses. A user manipulates the clock on a playback device so that the time-based license expiration is reached later than it should (or not at all). To detect clock roll-back, time is sampled on the client device **10** when the application is registered and every time it starts up, and the time is stored into the encrypted file. When a player is instantiated to play a media item, a separate thread is also started to monitor the progression of time during playback. The thread sleeps for a short time period, wakes up, and increments an elapsed time counter. That elapsed time is added to the last known local time. Thus, the application always has information about what the time should be (to an approximation). This technique can be augmented to include time information from a server **12**.

Rights File Integrity Protection

[0067] The rights file (also referred to as rights object herein) is stored on the client device **10** and contains the device-specific key and the content-keys encrypted with the

device-specific key. The rights file itself is encrypted with the key constructed from the domain key and the unique identifier of the device. The contents of the file are checksummed and the checksum itself is stored within the file. When the file is decrypted, the contents are checksummed again and the computed checksum is compared with the checksum stored in the file to verify that the file has not been tempered with. The rights file also has a copy protection feature, in a sense that an outdated copy of the file cannot be written over the fresh copy without being detected by the DRM Agent. The copy protection is platform-dependent. On the iPhone/iPad platforms, DRM Agent obtains a unique property of the file and stores it within the encrypted file. The unique file value is not something that can be controlled at will, it is a property that is assigned by the operating system. Those skilled in the art may choose this file property such that copying the file would force a change in the unique value. On Android the rights file is stored within the application-specific directory which is protected from other applications and from user access via standard Linux permissions. Furthermore, DRM Agent generates a random long number and stores it within the encrypted file as well as within the application-specific directory on the device. The two numbers are compared when mobile application starts. On the Blackberry platform, a similar randomly generated long number is stored inside the encrypted file as well as within the application-specific persistent secure storage offered by the Blackberry platform.

[0068] In the description herein for embodiments of the present invention, numerous specific details are provided, such as examples of components and/or methods, to provide a thorough understanding of embodiments of the present invention. One skilled in the relevant art will recognize, however, that an embodiment of the invention can be practiced without one or more of the specific details, or with other apparatus, systems, assemblies, methods, components, materials, parts, and/or the like. In other instances, well-known structures, materials, or operations are not specifically shown or described in detail to avoid obscuring aspects of embodiments of the present invention.

[0069] Although the above description includes numerous specifics in the interest of a fully enabling teaching, it will be appreciated that the present invention can be realized in a variety of other manners and encompasses all implementations falling within the scope of the claims herein.

What is claimed is:

1. A system for protecting media items delivered over the Internet to mobile devices wherein a license to each media item for a given mobile device is individualized to the given mobile device.

2. The system of claim **1** wherein the delivery is via segmented files over a hypertext transfer protocol.

3. The system of claim **1** wherein the license is also individualized to a specific user.

4. The system of claim **1** wherein the media item may have a plurality of representations for different bitrates.

5. The system of claim **4** wherein the player may switch arbitrarily between bitrates.

6. The system of claim **1** wherein the content is played offline.

7. A method for protecting content to be delivered to client devices via a content delivery network, comprising:

creating a media encryption key on a per-media basis during ingestion of media;

encrypting the media using the media encryption key;

pushing the encrypted media to the content delivery network for later delivery to client devices.

8. A method according to claim **7**, further including per-media configuration of specific distinct encryption ciphers on a per-media basis.

9. A method according to claim **7**, wherein the encryption ciphers include one or more stream ciphers for which a final key length is also per-media configurable.

10. A method according to claim **7**, wherein encrypting the media uses one or more encryption ciphers selected from advanced encryption standard (AES), RC4, HC-128 and XOR.

11. A method according to claim **10**, wherein encrypting the media includes performing an XOR operation using a key of a certain length L combined with data of protected content, the XOR operation comprising:

exclusive-OR'ing the key with L content bytes starting at a selected position of the content file;

exclusive-OR'ing the key with subsequent sets of L content bytes of the content file until all content bytes have been XOR'ed with the key.

12. A method according to claim **11**, wherein the selected position is a predetermined position at or near a beginning of the content file.

13. A method according to claim **11**, wherein the selected position in a beginning portion of the content file and chosen just prior to encrypting the content file.

14. A method according to claim **11**, further including:

downloading an encrypted media item to a client device;

decrypting the encrypted media item only during playback and only in small quantities necessary for immediate playback, each small quantity spanning a range of content bytes determined by operation of a video player performing the playback and also by user scrubbing actions during playback, the scrubbing actions including rewinding and fast forwarding, the decrypting further including receiving a byte range and an offset of a first byte in the byte range from the beginning of the encrypted , and computing a mapping between the key and content bytes which need to be decrypted.

15. A method according to claim **10**, wherein encrypting the media includes performing an HC-128 operation using a key of a certain length combined with data of protected content, the HC-128 operation comprising:

initializing an HC-128 stream cipher algorithm and generating a stream of bytes of a preconfigured length, the length being configured on a per-media basis, the initialization including setting the value of an initialization vector employed by the HC-128 stream cipher algorithm to zero;

for a first set of content bytes of the preconfigured length, exclusive-OR'ing the stream of bytes with the set of content bytes;

for subsequent sets of the content bytes, (1) incrementing the initialization vector, (2) re-initializing the HC-128 stream cipher algorithm with the incremented initialization vector, and (3) repeating the generating of the stream of bytes and the exclusive OR'ing the stream of bytes with the subsequent sets of content bytes until the entire content file is fully encrypted.

16. A method according to claim **15**, further including:

downloading an encrypted media item to a client device;

decrypting the encrypted media item only during playback and only in small quantities immediately necessary for

playback, each small quantity spanning a range of content bytes determined by operation of a video player performing the playback and also by user scrubbing actions during playback, the scrubbing actions including rewinding and fast forwarding, the decrypting further including (1) receiving a range of bytes located anywhere within the media item along with an offset from the beginning of the media item, (2) identifying one or more initialization vectors needed to decrypt the bytes, (3) initializing the HC-128 algorithm using one of the identified initialization vectors, (4) generates a key stream, (5) exclusive-OR'ing the key stream with the bytes, and (6) repeating the above steps (3), (4) and (5) using distinct other ones of the initialization vectors for subsequent ranges of content bytes as necessary until the media item is fully decrypted.

17. A method according to claim **10**, wherein encrypting the media includes performing an RC4 operation using a key of a certain length combined with data of protected content, the RC4 operation comprising:

initializing an RC4 stream cipher algorithm and generating a stream of bytes of a preconfigured length, the length being configured on a per-media basis, the initialization including setting the value of an initialization vector employed by the RC4 stream cipher algorithm to zero;

for a first set of content bytes of the preconfigured length, exclusive-OR'ing the stream of bytes with the set of content bytes;

for subsequent sets of the content bytes, (1) incrementing the initialization vector, (2) re-initializing the RC4 stream cipher algorithm with the incremented initialization vector, and (3) repeating the generating of the stream of bytes and the exclusive OR'ing the stream of bytes with the subsequent sets of content bytes until the entire content file is fully encrypted.

18. A method according to claim **17**, further including:

downloading an encrypted media item to a client device;

decrypting the encrypted media item only during playback and only in small quantities immediately necessary for playback, each small quantity spanning a range of content bytes determined by operation of a video player performing the playback and also by user scrubbing actions during playback, the scrubbing actions including rewinding and fast forwarding, the decrypting further including (1) receiving a range of bytes located anywhere within the media item along with an offset from the beginning of the media item, (2) identifying one or more initialization vectors needed to decrypt the bytes, (3) initializing the RC4 algorithm using one of the identified initialization vectors, (4) generates a key stream, (5) exclusive-OR'ing the key stream with the bytes, and (6) repeating the above steps (3), (4) and (5) using distinct other ones of the initialization vectors for subsequent ranges of content bytes as necessary until the media item is fully decrypted.

19. A method according to claim **10**, wherein encrypting the media includes performing an AES operation using a key of a certain length combined with data of protected content, the AES operation comprising:

randomly generating AES encryption keys on a per-media basis at the time of ingestion of the media;

representing the media as multiple segments having corresponding sequence numbers;

establishing distinct initialization vectors for the segments;

performing an AES encryption algorithm on each segment using the respective initialization vector, the AES encryption algorithm including cipher block chaining within each segment and not spanning multiple segments.

20. A method according to claim **19**, further including:

downloading an encrypted media item to a client device;

decrypting the encrypted media item using a native player of the client device, the native player having access to a playlist specifying a content decryption key, the playlist being served from a local HTTP server executing on the client device, the key being delivered to the client device from a digital rights management server via an encrypted message.

21. A method for delivering protected content to a client device, comprising:

registering the client device;

obtaining the license for use of the protected content;

decrypting the protected content; and

playing the decrypted content using a player of the client device.

22. The method of claim **21**, further including detecting clock roll-back by use of a DRM agent executing on the client device to:

monitor a local clock to ensure that a time reported by the local clock is continuously incrementing;

at a time of starting a playback application, sampling the time and comparing it to a last known time stored in an encrypted rights file on the client device; and

employing a separate thread which wakes up at regular intervals as playback proceeds and (1) keeps track of elapsed playback time, (2) adding the elapsed playback time to the previously sampled local time on the client device, and (3) writing the result to the encrypted file.

23. The method of claim **21**, further including detecting content expiration by:

maintaining rights information including an expiration time and/or maximum play count in association with each media item, the right information being received from a digital rights management (DRM) server in an encrypted form, and storing the rights information in an encrypted file on the client device;

upon initiating playback of the media item, employing a DRM agent on the client device to determine whether the media item has expired by comparing a current time on the client device to the expiration time of the media, and/or comparing the maximum play count to a stored actual play count, the DRM Agent ensuring that clock rollbacks on the client device are detected and disallowing media playback if time is not steadily incrementing, the DRM Agent executing a separate thread, in parallel with a thread performing playback such that the media item can expire not only upon initiating playback but also during playback.

24. The method of claim **21**, further including

uniquely identifying the client device using a unique physical device identifier, the unique physical device identifier being obtained from the client device using an application programming interface offered by an execution environment of the client device, the unique physical device identifier being encrypted with a domain secret and stored in an encrypted file on the client device and also sent to a digital rights management (DRM) server via an encrypted message, the encrypted unique physi-

cal device identifier being stored by the DRM server in a database of registered client devices and being mapped to a unique logical device identifier representing a user of the client device, the unique logical device identifier being sent encrypted with the domain secret.

25. The method of claim **24**, wherein the unique physical device identifier is selected from an IMEI number, an MEID, an ESN and a UDID.

26. The method of claim **24**, further comprising:

combining the unique physical device identifier with a domain key to generate a device-specific key to encrypt a rights file stored on the client device and containing content decryption keys used to decrypt the protected content.

* * * * *